Pers Ubiquit Comput
DOI 10.1007/s00779-014-0778-z

ORIGINAL ARTICLE

Persuasive software design patterns for social influence

Michael Oduor · Tuomas Alahäivälä · Harri Oinas-Kukkonen

Received: 1 November 2013 / Accepted: 12 June 2014

Abstract This article describes software design techniques for social influence as software design patterns, instantiating social influence features defined in the persuasive systems design (PSD) model. The article draws on literature from PSD, social psychology and software patterns to derive the social influence patterns, which are then implemented in a conceptual system. This paves the way for an important research track within persuasive systems research. The ultimate aim of these persuasive software patterns was to develop generalizable techniques that could aid the development of social support features in any persuasive system.

Keywords Software patterns · Persuasive systems design · Social influence · Behavior change support systems · Persuasive technology

1 Introduction

Information technology is never neutral and influences people's attitudes and behaviors in one way or another, and this influence may at times be an unintentional side effect of the design [45]. In this regard, persuasive technology describes how interactive computing systems have an impact on users' thoughts and consequently lead to a change in their behavior [19, 45]. Building on from persuasive technologies are behavior change support systems (BCSS) [42, 43]: Socio-technical systems focusing more on users' goals and concerned with both psychological and behavioral outcomes. Hence, designing systems that aim at behavior change require a thorough understanding of the problem domain and the underpinning theories and strategies of persuasive systems design (PSD) [32].

Human beings usually acquire particular habits and other behavioral patterns resulting from how they have constantly behaved, consequently leading to development of either a positive or negative self-concept, which affects their ability to learn, interact, develop and change [3–5]. Moreover, in creating persuasive experiences via technology, how users behave plays an important role in determining whether the system works as it should and the best scenario is when people use the system in the intended way—although unintended use may provide learning opportunities that may further be integrated in future iterations of the system [34]. For social support features to be effectively applied in persuasive systems, an understanding of fundamental aspects of human behavior is required. Additionally, mediums for channeling this understanding in stimuli that enhance positive user experiences and interaction via the persuasive systems need to be devised. This will ensure that users are aware of the key software features, perceive them as intended by system designers and know how the use of these features will guide their ongoing interactions.

In current persuasive system research, which is driven by theories on persuasion, there seems to be a tendency of describing the software systems and the persuasion context at too general a level because of a lack of guidance for software developers of persuasive systems [42, 43, 60] with relatively few works putting the emphasis on describing the

M. Oduor (✉) · T. Alahäivälä · H. Oinas-Kukkonen
Department of Information Processing Science,
University of Oulu, Oulu, Finland
e-mail: Michael.Oduor@oulu.fi

T. Alahäivälä
e-mail: Tuomas.Alahaivala@oulu.fi

H. Oinas-Kukkonen
e-mail: Harri.Oinas-Kukkonen@oulu.fi

Published online: 03 July 2014

actual software system clearly [cf. 1, 51]. Additionally, in most of the research concerning BCSSs and the broader topic of persuasive technology, there are theories intended for behavior change, but these theories do not provide guidance for the implementation of systems that support behavior change, which easily leads to a passive adoption of these theories for systems implementation and into a mismatch between the persuasive message and the selected strategy [23, 42, 60]. Black-box thinking of the software systems with no actual description of what was implemented and how may make the research results obsolete. Hence, persuasive software design patterns have been suggested as a new avenue of research into persuasive design to tackle this problem [42].

Formulation of software design patterns can help distinguish between various social support features and their application in systems development—paying particular attention to the intent. This is because software design patterns are reusable solutions that can be applied to commonly occurring problems in software design, and they enable building of systems with good object-oriented design qualities that can be easily understood, maintained and revised [22]. That is, systems expressed in terms of Classes and Objects representing real-world entities that have attributes (variables) and functionality (methods) can be acted upon. An Object is an instance of a Class, for example, a Person is a Class, and Homer is an Object of type Person who has the attributes name, date of birth (DOB), social security number (SSN) and the functionalities of sleep, read, talk, etc. [22]. Patterns guide and are also discovered (developer uniqueness may lead to different outcomes in overall design) as the development process progresses, and they primarily address issues concerning changes in software.

Therefore, the purpose of this paper is to develop software design patterns for social influence by examining the effect(s) of technology on persuasion—how users' thoughts and actions are influenced—and easing the behavior change process. Additionally, since there have been inconsistencies in the classification of social influence constructs in information systems (IS) research [63] and as these features have been categorized in Oinas-Kukkonen and Harjumaa's [45] PSD model, developing patterns based on the principles are a potential solution to aid in addressing the issue. Consequently, the patterns can be considered as instantiations of the social support features in the PSD model that extend the principles from design and evaluation into actual software implementation, which is requested by Wiafe et al. [60] for example. Thus, software design patterns to be utilized in development of actual systems are formulated.

These patterns are intended to serve as guidelines and provide a reusable format that can be applied in future systems development. In order to develop the patterns, relevant literature that highlights the characteristics and issues to consider in PSD [18, 32, 36, 45], social influence [16, 25, 39, 55] and software design patterns [14, 24, 28, 56] will be reviewed.

The article is structured as follows. Chapter 2 provides an overview of PSD and social psychological principles from which we derived the suggested patterns. Chapter 3 presents the existing literature on software design patterns, forming the basis for pattern development and serves as an introduction to Chapter 4, which integrates literature on influence and pattern development to develop the patterns for social influence. This is then followed by the discussion and conclusion that summarize the preceding chapters and identifies the future directions.

2 Theoretical background

2.1 Persuasive systems

Computers were originally created for performing rudimentary tasks like calculating, storing and retrieving of data; they have now adopted increasingly persuasive roles as they have shifted to our everyday lives as a result of their ubiquity [19]. In terms of persuasive technology, computers are viewed as interactive technologies that can motivate and influence the user. Oinas-Kukkonen and Harjumaa [44] define a persuasive system as "*a computerized software or IS designed to reinforce, change or shape attitudes or behaviors or both without using coercion or deception.*"

Three potential outcomes for a persuasive system are reinforcement, i.e., making current attitudes resistant to change, changing outcome, i.e., relating to one's response to an issue and shaping outcome, i.e., formulating a pattern for situations where one did not exist before [44]. Technology does not in itself seek to influence, but through services that can be built on top of it, it facilitates behavior change also simplifies the behavior change process [36]. Persuasive communication therefore aims at voluntarily changing one's behavior, and for persuasion to be effective, it should also be based on the voluntary participation of the user(s) and their empowerment to change target behavior in the persuasion process [19, 43].

Brinol and Petty [6] outline persuasion as a situation where a recipient receives an intervention (e.g., a persuasive message) from the sender referring to something outside the message itself (i.e., the context). Persuasion techniques are most effective when they are interactive and when persuaders adjust the influence approach as the situation changes. Computers can be more effective than humans in persuasion because they are more persistent,

they offer greater anonymity, they can manage huge volumes of data, they can present information in numerous modes—modality (such as data and graphic, rich audio and video and animation), which humans cannot match, they can scale according to demand, and finally, they are ubiquitous—they can be accessed from almost anywhere and at any time [19].

To a growing extent, we live a nomadic life and own devices that we carry about with us [46], so an interactive computing system that automatically adjusts to the prevailing environment (the processing, communications and access requirements needed at a particular place), and users' computing needs in an integrated way is essential [27]. Such a system—especially in the case of interactive computing systems that are embedded in our everyday objects—enables intervention at any time (ideally, at the opportune moment) and place. This increases the chances of getting results by enhancing their persuasive ability to motivate through enhanced connectivity and mobility [19] as they are focused on helping users to achieve their goals [50]. The multi-modality and high interactivity of web technologies enable creation of self-regulatory skills and also provide various options for engaging, educating and equipping individuals [37].

Research into attitude change in recent years has predominantly been guided by Petty and Cacioppo's [48] Elaboration likelihood model (ELM), whose two underlying principles are the factors influencing attitude change, depends on one's motivation and ability to process relevant information. Elaboration refers to the extent of scrutiny of a persuasive message and its arguments and "*the likelihood of elaboration will be determined by a person's motivation and ability to evaluate the communication presented*" [48]. The model proposes two relatively distinct routes to persuasion: the peripheral route—which are simple cues one relies on when they are not highly motivated and/or have low ability—and the central route—where a person is motivated and able to carefully and thoughtfully consider the true merits of information presented in support of a relevant proposition. In the latter case, attitude change will be more enduring and act as a predictor of behavior [48].

Central route to persuasion emphasizes argumentation, consistency and credibility, whereas the peripheral route relies on simple cues (or rules of thumb), for example, "rare = valuable," to categorize things based on a few key features, which determines ones response [44]. Both routes may be supported through numerous persuasion techniques, one of which is Cialdini's [12] "weapons of influence": Reciprocity—repaying in kind what we have received from someone else; commitment and consistency—the desire to be consistent with our previous actions; social proof—deciding on what is correct by finding out others' thoughts; liking—pressure to say yes to someone one likes because of the tendency to associate a good quality (e.g., looks) with other similar qualities; authority—obligation to obey and listen to a person with a higher status; and scarcity—things seem more appealing when less available [12, 44]. The ELM states that personally relevant information—or those perceived as such—increases motivation to elaborate and consequently, ones receptivity to persuasion attempts.

According to Fogg's [20] behavior model, behavior is a product of three factors, namely motivation, which is required to perform a target behavior, ability, which is required to perform the behavior, and triggers—stimuli to perform the behavior. In order for an intervention to be successful, it thus has to be noticed, associated with an event and it should occur when one is both motivated and able to perform the required behavior [18].

Humans often rely on intuition to achieve target behaviors, something which computers cannot, yet rely on to create persuasive experiences. For efficiency, these experiences must be pre-coded in the systems and thought must be given to the target behaviors desired and the means of achieving those [17]. From a user's perspective, computers can operate as tools that increase capability, as mediums that provide interactive experiences, and as social actors that create—and enhance—relationships [19]. Fogg's [19] work builds on Nass et al.'s [40] proposition of computers as social actors and that individual's interactions with computers are primarily social. Through experiments where social norms and notions of "self" and "other" were applied to computers, it was deduced that people's responses to computers are not the result of conscious beliefs that computers are human or possess human-like features, of ignorance, psychological dysfunction or the belief that the computers were acting as mediators. Rather, it is because it is common and relatively easy to produce social responses to computers [40].

Extending Fogg's [19] studies on persuasive technology is Oinas-Kukkonen and Harjumaa's [45] PSD model for designing and evaluating persuasive systems, which categorizes persuasion techniques according to four design principles based on aspects of human behavior which need to be recognized in order for persuasion to be effective: (1) Primary task—that supports the carrying out of a user's real-world activity and is comprised of reduction, tunneling, tailoring, personalization, self-monitoring, simulation and rehearsal. (2) Dialogue support—deals with degree of feedback and features that support interaction between the users and the system and is comprised of praise, rewards, reminders, suggestion, similarity, liking and social role. (3) System credibility support—deals with designing more credible and subsequently more persuasive systems and is comprised of trustworthiness, expertise, surface credibility, real-world feel, authority, third-party endorsements and

verifiability. (4) Social support—deals with designing systems that motivate users by leveraging social influence. This includes social facilitation, social comparison, normative influence, social learning, cooperation, competition and recognition [45].

In line with the purpose and the focus of the article (on social support features), the ELM and other principles discussed above will be applied in helping to establish the content (of persuasive messages) and determine the appropriate method for message delivery. As the social support category is the basis for development of the patterns, we now expound on some of the principles in this category and details of the specific patterns are provided in the following chapters (Chapter 4).

2.2 Social influence

Research in social psychology has shown that persuasion is predictable; humans have a psychological desire to express their identity, opinions and relationships [59]. Persuasive systems create tools that enable people to express themselves thus tapping into the deep-rooted human desire for expression and interaction [59]. In persuasive communications, people are sensitive to the frequency of occurrence of a behavior and are often able to—without prior thought—come up with predictions about the percentage of other people engaging in behaviors ranging from, for example, cooking to greed and normative beliefs about the occurrence of a behavior are often correlated with the person's own performance of that behavior [41]. Facing the same situation, different people may react differently. What guides our behavior is less the situation as it is than the situation as we construe it [5]. And as people do not live their lives in isolation, collaboration is required to attain what they cannot accomplish on their own [5]. Additionally, people's analysis of their public statements can cause shifts in privately held attitudes to match their public acts—impact of behavior on attitudes [61].

According to Guadagno and Cialdini [25] within the social influence discipline, there are two extensive areas: compliance—examining behavior change and mainly dealing with normative influence and persuasion—examining changes in attitudes and beliefs [25]. Social influence is common in our daily activities where we either try to influence or are influenced by the actions of others numerous times a day, and the influence arises from social situations, for example, in relatively trivial issues like the choice of a diner or more significant ones such as choosing to join a campaign for better pay or to raise money for humanitarian aid [53].

The opinions we hold about subjective matters are often the reflection of the opinion held by others, and these are often—directly or indirectly—influenced by cultural norms, mass media and interactions on social networks [39]. Social influence is defined as a change in one's beliefs, behavior or attitudes caused by external pressures that may be real or imagined [25]. When computers are perceived as social actors, they can leverage the above principles to motivate and persuade [19]. The behavioral goals of social influence recipients—which may not always be conscious decisions—are managing the self-concept, fostering and preserving relationships and acting effectively, and the processes involved in group interactions can change behavior and attitudes [9–11, 53].

Although it may not be clearly discernible from a user point of view, it has nowadays become common to combine multiple behavior change principles [52] such as self-monitoring and social comparison in persuasive systems, which can expose users to numerous motivating or stressful influences. Integrating these various behavior change principles increases the influence over the user and enhances the possibility for behavior change, but also exposes the user to many influences, which may be confusing [52].

Furthermore, people who are frequent in their online communities are not just seeking information, but they would also like to meet other people, get social support, friendship and a sense of belonging, i.e., they attempt to develop social relationships [8]. As studies on the effect of behavior on attitudes have shown, people's interpretations of their public interactions and other attitude-concerned behaviors can initialize changes in privately held attitudes to be consistent with public acts [61]. Asch (1940) "argued early on that the primary process in influence is not a change in attitudes toward an object but rather a change in the definition and meaning of the object—when meaning changes, attitudes change accordingly" [as referenced in 61].

In summary, there have been numerous studies that have applied psychological theories—with varying results—such as those on influence discussed above in both practical and technical interventions for health and sustainable living as these theories provide relevant background information and inspire the design and development of new constructs. The theories implementation especially in computer science, software engineering and IS research continues to evolve as the field matures and new inventions (e.g., the growth of online social networks, location-based presence applications) that revolutionize the approach to communication, work and relationships are made. A common approach to enhance research is to determine the usefulness of these theories when combined with concepts from other disciplines, hence the need to highlight their characteristics. This is in line with the stated purpose in the previous section and the patterns proposed and developed in the following sections. We combine principles from PSD and social influence with those from software design patterns to formulate the persuasive patterns.

3 Software design pattern thinking

Approaches to idea generation and problem solving have emphasized the process of recognizing that a new problem situation is similar to one faced (and solved) previously elsewhere, even in a different discipline [35]. Design patterns solve design problems by finding appropriate objects, determining object granularity, specifying objects interfaces and implementations, putting reuse mechanisms to work and designing for change [22]. Much of the current work on patterns stems from Alexander et al.'s (1977) work on covering the design and layout of buildings, towns and communities. In software engineering and human—computer interaction, patterns do not just describe recurring situations and their solutions, but they describe a method for presenting the situation and/or solutions in a structured way [as referenced in 35]. The patterns presented by Alexander et al. (1977) were adapted to object-oriented programming in the 1970s, and this was further adopted into software engineering in Gamma et al. [22], which is to-date among the mostly commonly referenced works on the topic.

Gamma et al. [22] classified patterns according to purpose and scope. The former refers to what a pattern does, and the latter specifies whether the pattern applies primarily to classes or objects. Under the purpose, patterns can either be creational, structural or behavioral. Creational patterns deal with initializing and configuring classes and objects—process of object creation. Structural patterns deal with the composition of classes or objects and identify simple ways to realize relationships between different sets of classes and objects. They help limit the scope of changes to only the section required for the change. Behavioral patterns deal with dynamic interaction and distribution of responsibility among classes and objects [22].

Patterns depict the static and dynamic structures and collaborations of successful solutions to problems—discerning of non-functional features for example—that arise when developing applications within a particular context. Patterns (or their solutions) should be applicable in many different situations without the need to make extensive changes as they provide ways to arrange and categorize relatively mundane solutions in technology-related development projects. Patterns have four essential characteristics, which could be extended to cover more issues like the intent, motivation, applicability and structure that will be covered in the following section [22]:

> The pattern name—a common term that eases the communication among stakeholders and enables design at a higher abstraction level while simplifying thoughts on designs and communicating these and their trade-off to others.
>
> The problem describes when a pattern should be applied and its context.
>
> The solution provides an abstract description of a design problem and how a general arrangement of elements (classes and objects) solves it.
>
> The consequences are the results and trade-offs of applying the pattern.

Most of the studies on software design patterns—from the time the Gamma et al.'s [22] book was published to-date—have focused on different stages of system development depending on the goal of the particular author(s). For example, there have been studies on the challenges involved in software architecture evaluation and visualization of software systems as a set of design patterns and possible solutions through use of (UML) unified modeling language class diagrams to document a system's static structure [33, 57]. Diaz et al. [14] visually represent web design patterns for end users by integrating them with goal-oriented design to assist both end users and casual developers in selecting the right patterns needed for specific design projects, determining the complexity of the project and determining the relationships and trade-offs of each.

Others [2, 28, 47] have focused on the various types and changes of patterns in the software development process where each of the stages—requirements elicitation and specification, design, coding, implementation through to final testing/debugging—has their own styles in the artifact change process that could be reusable in other development processes. Software patterns can be considered in each of these steps. Franch [20] for example, has particularly concentrated on patterns in the requirements phase of system development, and in this phase, there could also be interview and questionnaire patterns, in the design phase architecture and design patterns and so forth. The first step is choosing a suitable pattern, instantiating it to make it adaptable to the problem in order to be able to get an artifact. Arvesano et al. [2] in their empirical study conclude that the pattern change frequency and amount of co-change do not depend on the pattern type, but rather on the role played by the pattern to support the application features. Patterns are often changed either in their implementation or by adding subclasses or changing method interfaces: The latter, however, causes a higher co-change on client classes [2].

Zhu [64] and Pena-Mora and Vadhavkar [47] detail application of patterns in software reuse where an effective method of software development based on pattern reuse is proposed. Zhu [64] further subdivides software patterns into three granularity levels—architectural patterns, design patterns and idioms—and states that development based on reuse of software patterns has numerous advantages. In both of these works, a framework that provides valuable

insights when using patterns for development and analysis of reusable software systems is provided. Gestwicki and Sun [24] show how through game development design patterns could be taught with emphasis on object orientation and patterns integration. The study provides relevant examples of how patterns could be used to implement various features of a system's architecture—determining the relevant features required (domain-specific problem) leads to justification for the design patterns, which are presented with class diagrams using UML collaboration notations [24].

Additionally, there are some articles which mainly concentrate on architectural features such as [31, 58], which detail the use of the model-view-controller (MVC) architectural pattern in Web IS development and how Leff and Rayfield's [31] proposed model of flexible web application partitioning can be implemented using the constructs of MVC. Phek et al. [49] discuss on how web applications can be improved by analyzing two architectural patterns, MVC and presentation-abstraction-control (PAC), and five navigational design patterns—dealing with visual features—with the goal of discerning the most suitable design solutions for general features in web applications. Taleb et al. [56] provide a classification of different web design patterns and discuss how they can be combined in the development process while giving examples on their use and showing how the patterns have been used in the redesign of a website.

From the above discussion, it can be seen that patterns focus on most aspects of systems development and they can also be used in the various stages of development while providing reusable solutions and design guidelines for commonly occurring issues. Furthermore, most patterns allow some part of a system to vary independently of others, their use provides a shared language that maximizes the value of communication among developers, and reduces the time spent on making design decisions related to feature changes, and patterns enable reuse of solutions that have previously been effective [22]. And according to Buschmann et al. [7], currently, there is a common set of well-known software patterns, but when looking toward future developments, patterns could be more domain specific and tailored to particular focus areas, such as software-oriented architecture (SOA), mobile systems, Web 2.0, business process modeling, service orchestration, diverging protocols—yet to be properly covered by the pattern languages. Furthermore, as patterns do not yet cover all areas of group interaction in online environments [7], there are opportunities for development of persuasion-specific software patterns that could provide a proper basis for enhancement of social features in persuasive systems and more so in BCSSs.

The above discussion on patterns lays the foundation for the development of software design patterns for social support and provides an overview on the current and ongoing work on patterns, a direction that shapes the creation of the said patterns and establishes a basis for enhancing the theory on PSD.

4 Proposed software design patterns

Prior social psychological research (also discussed in Chapter 2) has shown how the social conditions influence people's behaviors in multiple ways. The way we perceive other people engage in activities and behaviors affects our interpersonal motivations to act [3, 4, 38, 62]. The design principles of the social support category in the PSD model [45] suggest many ways to include features for inducing social influence in persuasive systems. As instantiations of these constructs, we present four patterns that can be utilized in BCSS development to extract these principles as software features: social learning and facilitation (SLF), competition (COM), cooperation (COO) and recognition (REC), with SLF pattern being the underlying layer. The social support principles of normative influence and social comparison are at least partially implicit in the other features; for example, competition and recognition features to some extent utilize social comparison and normative influence via showing numeric figures from other people's performance.

The diagram (see Fig. 1) is an example of the use of the patterns. It is represented as a class, which presents the static view of a system—in a conceptual BCSS and describes a particular aspect (social support) of the whole application. The user and the SLF pattern are the two main elements in the system with a one-to-many relationship because the user can perform multiple tasks. The SLF is an abstract class, which has the three subclasses (COM, COO, REC) with inheritance relationship, i.e., they have similar attributes and some additional operations. These patterns provide the structure, whereas the theories they are based on provide the content.

In order to demonstrate and verify our software design patterns for social influence, we used them to develop a non-functional software prototype for a conceptual BCSS. For this purpose, we chose to build a system for supporting the use of a motivational method called "don't break the chain."[1] It is based on the idea that for adapting behaviors, one should choose a habit and start performing it on a daily basis. Each day the habit is successfully kept up should be marked down to keep track of the amount of consecutive days the target behavior has been followed. Thus, the longer the "chain," the more resilient the user is to maintain the behavior. The concept serves as a

[1] http://lifehacker.com/281626/jerry-seinfelds-productivity-secret/all.

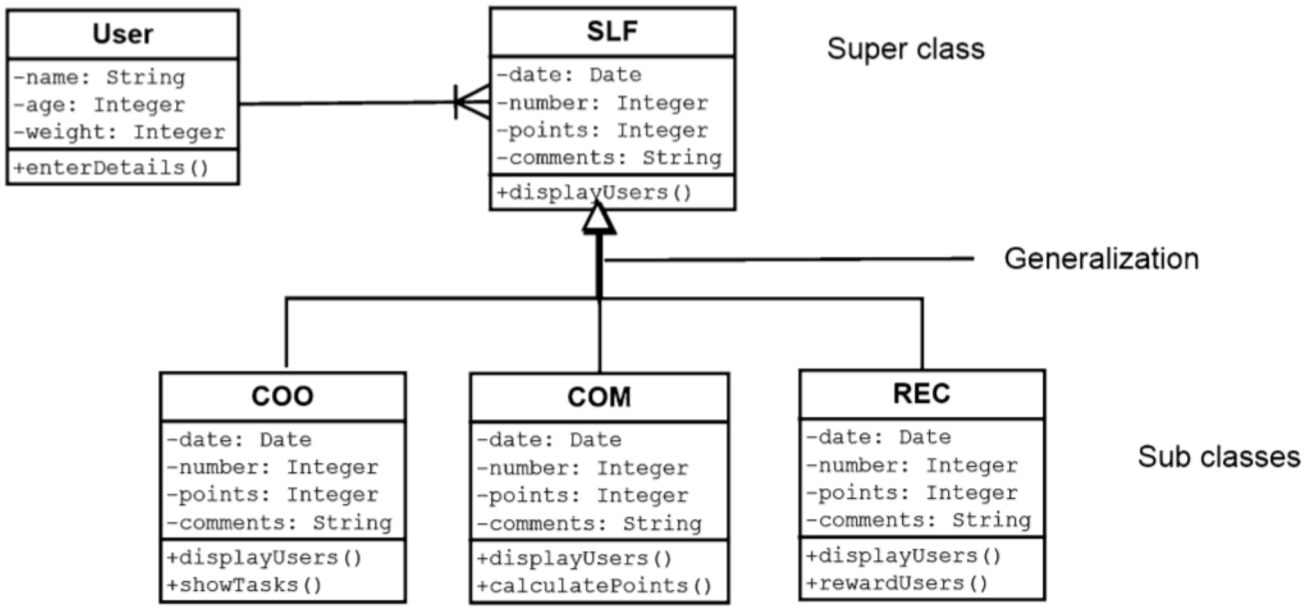

Fig. 1 Software design patterns for social influence

motivational technique to enhance commitment to the course and is suggested as an implementation in the system. Its principles are in line with and are also derived from the aforementioned theories above. The basic functionality of the system conceptualized here would be for the user to choose habits to follow and then keep marking down every day they maintain performing them. We now inspect how the system could be made more persuasive by using our presented software design patterns. For presentation purposes, we have omitted all but the features specifically implied by the design patterns in our prototype.

4.1 Software pattern: social learning and facilitation (SLF)

Social psychological research states that individual's behaviors are influenced in many ways by the behaviors of other people. The Social Learning Theory introduces the concept of observational learning, meaning that people's behavioral knowledge and skills are augmented through the act of observing others, having thus a direct influence on their own behaviors [3]. Another major motivational factor for behavioral change is social facilitation, based on the fact that people's awareness of being observed or evaluated by others also has a significant impact on their behaviors [62]. These concepts are present in the PSD model as the social learning and social facilitation principles in the social support category [45].

The social learning principle states a persuasive system should provide means for users to observe the behavior of their peers, whereas the social facilitation principle suggests users should be provided means for discerning other users performing the target behavior along with them. Experiments concerning the use of social influence features in a persuasive system have supported the claim that they may have multiple positive impacts on user perceptions and behaviors [52, 54, 55], and peoples' actions are influenced by the presence of others [21, 62].

As an instantiation of the aforementioned theoretical concepts and design principles, we posit that they can be instantiated in BCSSs by utilizing the design pattern of SLF. There exist other interpersonal motivating factors that arise from the conditions of social interaction, such as competition, cooperation and recognition, as presented by Malone and Lepper [38]. They actualize as emerged intrinsic motivation when functioning within the presence of a social setting. There can be numerous ways for making user behaviors visible to others within the system according to the SLF pattern, from which we thus present that the special cases of competition (COM), cooperation (COO) and recognition (REC) can be extracted as their separate design patterns (see Fig. 1).

Pattern Name and Classification social learning and facilitation (SLF).

Intent The main purpose of the pattern is to enable and enhance the design of software features that visualizes the presence of other people working toward a similar goal.

Motivation (Forces) It is easier for individuals to pursue set goals when there is a clear awareness of others facing the same issues and working toward the same. It is also good to consider individual differences and the prevailing life circumstances—the context—when setting goals as the

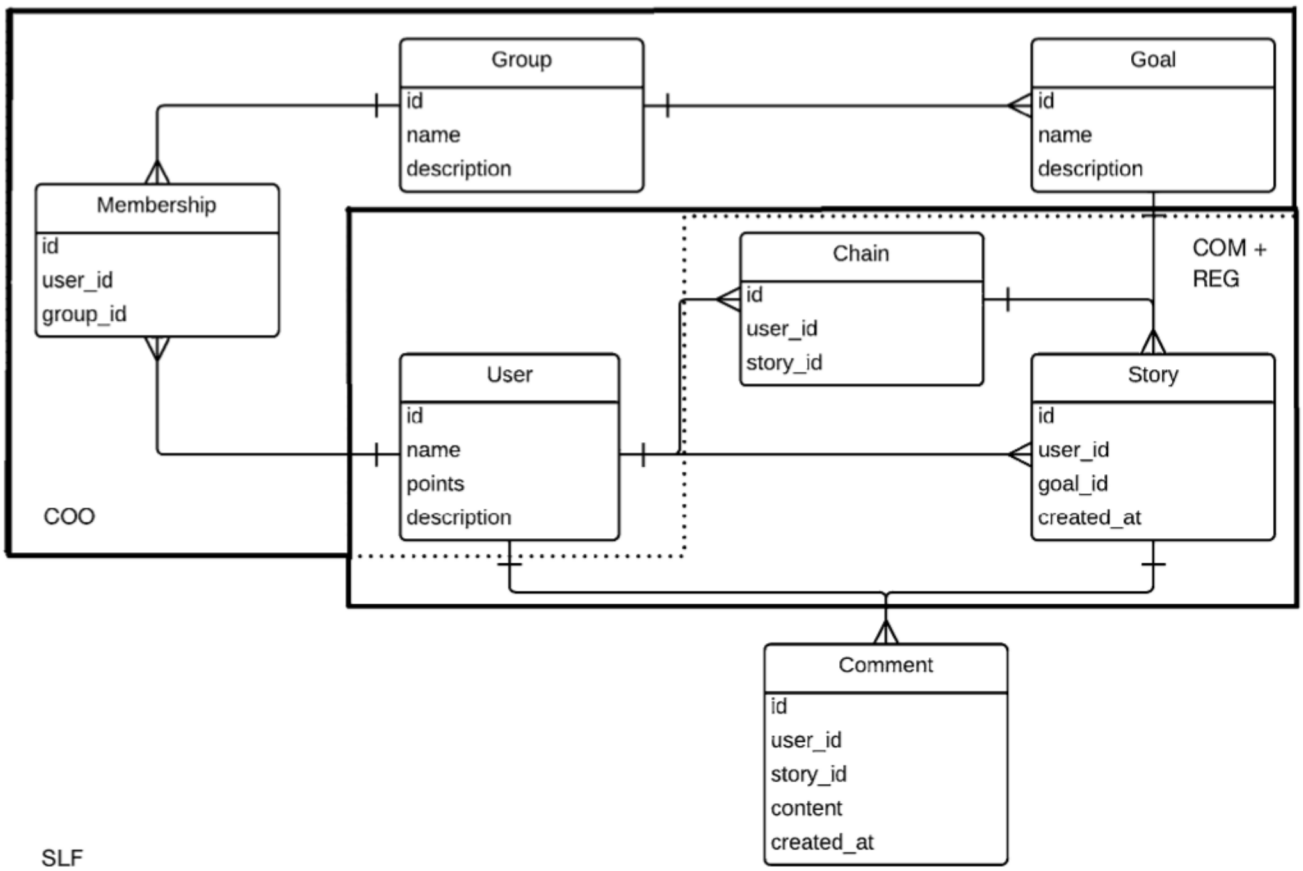

Fig. 2 Structure of the SLF pattern

context affects decision making (mostly unconsciously) as people are often not aware how the surrounding environment influences their decision making. The selection of habit when starting and the length of the chain as one progress' could be an indicator of the prevailing circumstances, and principles outlined in the ELM could be applied in determining the persuasion route—whether central or peripheral depending on the individuals' capabilities.

Applicability The *SLF* pattern states that the system should allow users to see the progress of their peers and make users aware so that others can observe their performance. An apparent way to achieve this would be to show user's action in the form of a newsfeed that is visible to everyone else using the system, or possibly to a limited audience.

Structure The SLF pattern is implemented (Fig. 2), as all the features in the system induce social interaction and enable perceiving other users. Essentially, they cover all the features that provide social influence, including the features implied by the subclass patterns. These include User, Chain, Story, Comment, Goal, Membership and Group. The connections of the components are represented via their connecting lines, some representing a one-to-many relationship. For example, a user can belong to many groups and have many comments.

Implementation Fig. 3 demonstrates the Newsfeed feature, which implements the *SLF* pattern. Mimicking the paramount feature of common social media services, it lists actions taken by the user and her peers within the system, making it possible to observe the progress of others in the system as suggested by the pattern. The actions may be commented upon, resulting in peer support and enabling further social learning within the system.

Known Uses Many of the contemporary exercise tracking applications such as Fitocracy,[2] Endomondo,[3] Runkeeper,[4] Sports Tracker[5] utilize this approach. Another possible implementation could be to show random profiles of other active users of the system, as done in Sports Tracker. An Android application known as Quit Smoking,[6] for example, compares one's smoking patterns against the

[2] https://www.fitocracy.com/.

[3] http://www.endomondo.com/.

[4] http://runkeeper.com/.

[5] http://www.sports-tracker.com/.

[6] https://play.google.com/store/apps/details?id=com.azati.quit&hl=en.

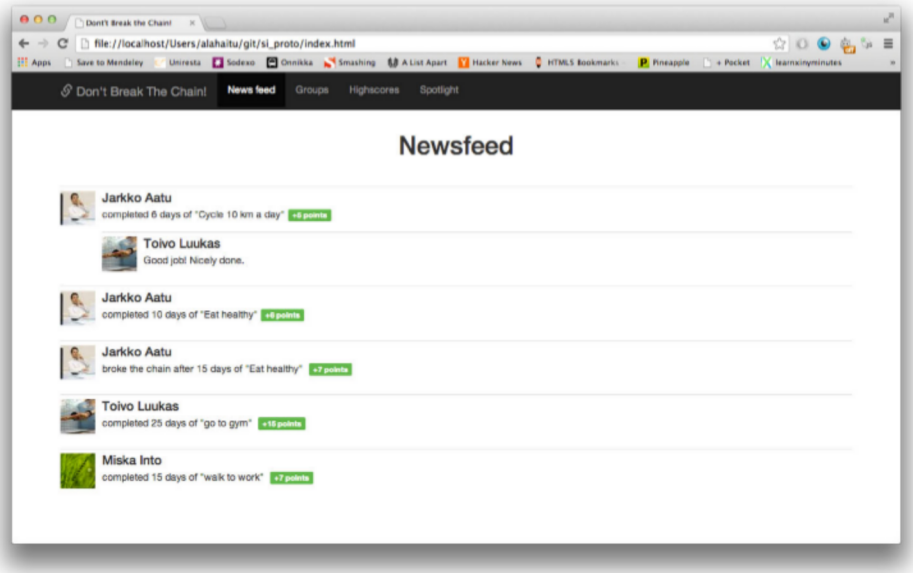

Fig. 3 Example implementation of the SLF software pattern

statistics of other users, which acts as a motivator. For the sake of protecting users' privacy, users should always be aware if their information is available to other users or user groups in which they do not belong.

4.2 Software pattern: competition (COM)

Competition is based on the judgmental process of self-regulation [4]. An example of competition and the effect it has on users is Lindqvist et al.'s [34] paper investigating why people use Foursquare. They find that representational systems such as Foursquare are progressively becoming connected with both our social and significant day-to-day practices. In the PSD model [45], competition is presented as a potential social support guideline to motivate users to adopt attitudes and behaviors by leveraging the people's natural drive for competition. Proceeding with this, we suggest the design pattern for competition as applying competitive elements, such as ranks, scores and levels that allow users to compare their performance with others' and adjust their goals based on that.

There exist many ways to implement the competition pattern as software features. Many of these can be discussed in pursuance of the emergent concept of gamification, implying the use of game elements in non-game contexts, or a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation [13, 26]. Competitive elements are not the only possible game-like features that may be used, but are clearly the prominent ones. It should also be noted that as individuals' personalities and attitudes toward competing may differ considerably, not everyone finds competition enjoyable. People could for example be wary of other people's reactions on them performing badly.

Pattern Name and Classification competition (COM).

Intent To design competitive features that function as motivational factors and result in competitive interactions among the users.

Motivation (Forces) Enabling of competitive elements such as ranks, scores and levels that allow users to compare their performance with others and adjust their target goals based on these. These competitive elements may function as additional motivational factors for users to engage in behavior change. But for some people, competition might be a source of anxiety, so participation should always be voluntary. In some domain areas like weight loss, competition might even be dangerous.

Applicability A real-life BCSS implementation of the Competition pattern would be the ranking of users in a high score lists based on their performances, such as levels and points received for engaging in the target behavior. This enables users to follow their natural competitive tendencies to reach a higher rank among their peers while pursuing their target behaviors.

Structure The COM is represented by the features of User, Chain, Goal and Story entities. The COM feature is implemented by encouraging users to gather points. Maintaining Chains that consist of Stories created at subsequent days adds to the points attributed to a User. Chain comprises of records of a user's subsequent Stories, and a Story itself always relates to a certain Goal (Fig. 4).

Implementation The high scores feature (see Fig. 5) demonstrate the implementation of the COM pattern. In our conceptual design, the users gain points each time they mark down a consecutive day of performing the habit. The

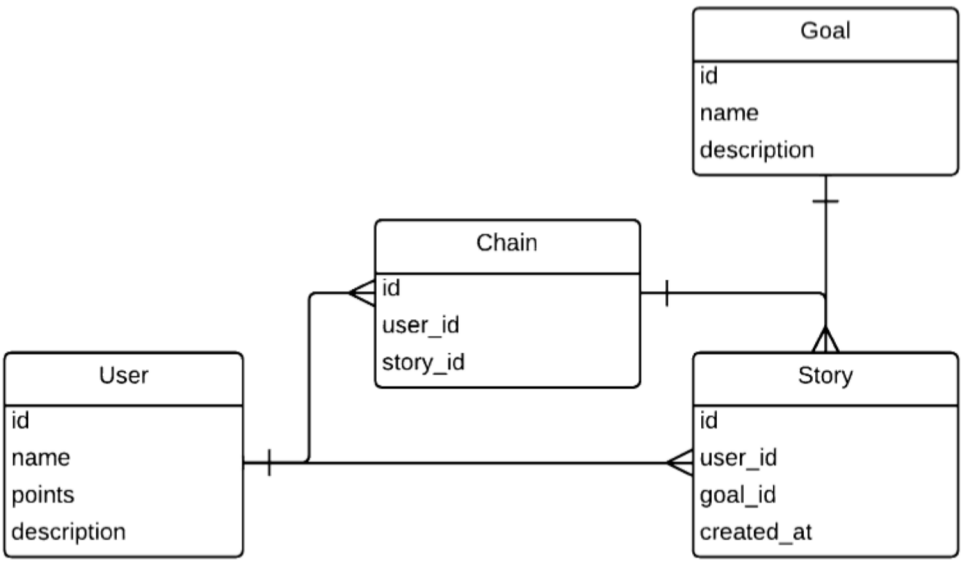

Fig. 4 Structure of the COM pattern and REC patterns

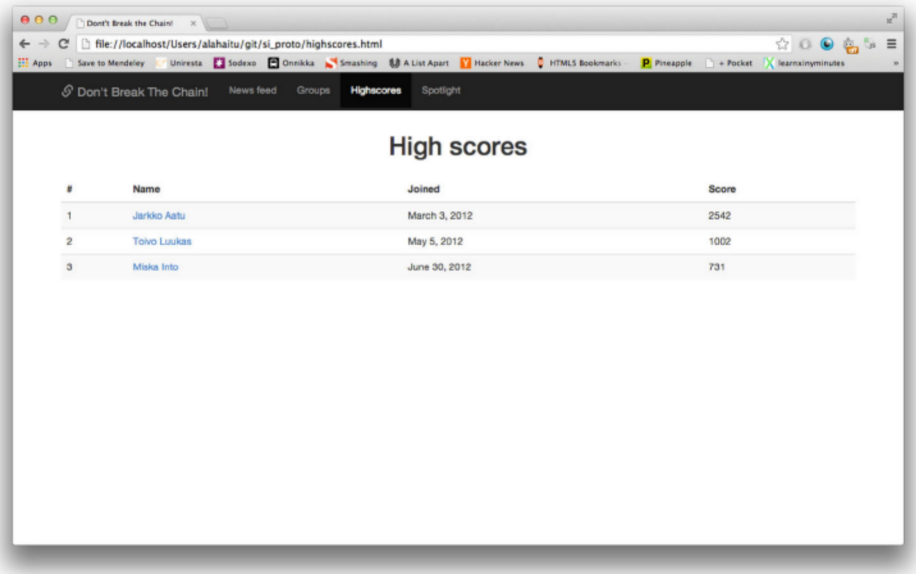

Fig. 5 Example implementation of the COM software pattern

points gained for each action are shown alongside it in the newsfeed. More points could be given for maintaining long "chains," based on a multiplier. The high score list then shows the users' rankings based on the points accumulated. This allows users to compare their performance with the others. In our example, we also show users the date they joined the service for them to be able to further interpret the results.

Known Uses Endomondo, for example, allows users to meet like-minded friends, get inspired and challenge them in mainly outdoor sports like cycling, walking, completing a run, etc. Fitsby[7] motivates a group of friends to reach a desired goal by asking them to contribute a certain amount of money—this is optional, and users can compete without monetary stakes—and the person who checks into the gym the most within a given time frame wins.

4.3 Software pattern: cooperation (COO)

In addition to competition, reciprocity appears to be apparent to human nature [12, 38], and cooperation is an interpersonal motivator that has proven to cause positive impact on user behaviors [54]. The PSD model suggests a system can motivate users to adopt a target attitude by leveraging this natural drive [45]. We present the design

[7] https://play.google.com/store/apps/details?id=com.fitsby.

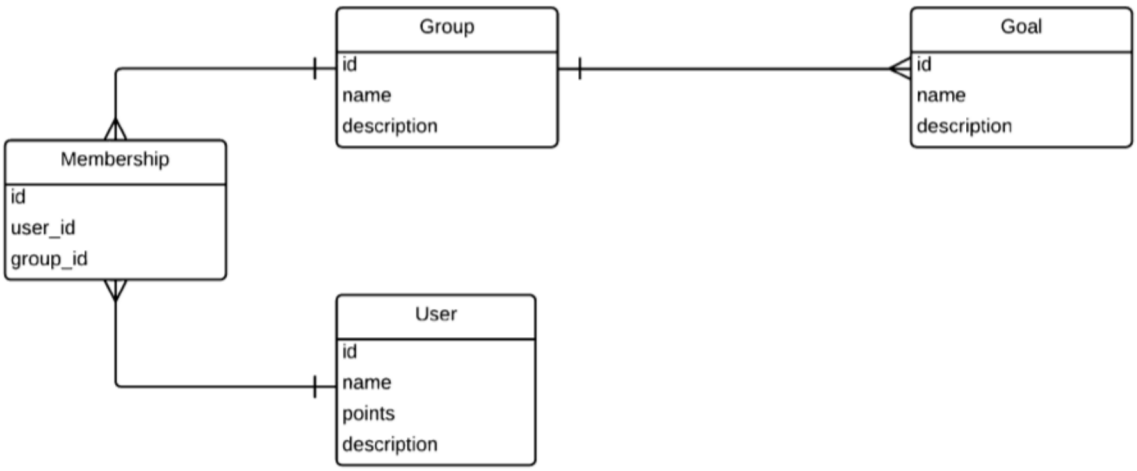

Fig. 6 Structure of COO pattern

pattern for cooperation in persuasive systems, which allow users to gather around mutual goals and support each other in achieving them.

Pattern Name and Classification Cooperation (COO)

Intent To design software features, which allow users to engage in mutual goals and provide ways for them to support each other in reaching their goals.

Motivation (Forces) It is easier to cooperate when there is a clear awareness of what needs to be done, what the other people are doing and how far they are in achieving their objectives. The system should be tuned to the users' needs and through analysis of their use provide relevant options for collaboration. Group discussion encourages giving and receiving peer support on the matter. Though, as some users prefer to perform alone, the pattern does not imply that cooperation is forced on them.

Applicability Social media services allow many ways for their users to cooperate. An evident example would be the possibility to establish virtual groups for like-minded people to communicate and support each other in achieving their goals.

Structure The COO pattern (see Fig. 6) can be implemented as the software components that allow users to form groups and collaborate by interacting with each other. In our example, the pattern is manifested as the User, Membership, Group, and Goal entities. User component has an aggregation relationship with the Group component, details of which are stored in the Membership entity. A group may include multiple goals. Membership holds records of users belonging to a group. Groups with mutual goals provide ways for cooperation within the system.

Implementation Designed using the COO pattern, the group page (see Fig. 7) enables users to share goals with their like-minded peers. There are different groups in the system based on mutual interest for behavior change, such as gaining better eating habits or exercising more regularly. By joining a group, the user can add goals that can be joined by other users, sharing similar ambitions. The amount of people engaging in a goal is shown in association with each goal. There could also be a discussion section in the group to encourage communication within its members.

Known Uses Fitocracy and other fitness applications encourage social interaction among its user and allow creating groups for people engaged in the same sports. Users can create challenges that the group members would like to participate in, and this enables the members to discuss only about their interests, what really matters for them in the achievement of personal and group objectives.

4.4 Software pattern: recognition (REC)

In addition to competition and cooperation, Malone and Lepper [38] introduce recognition as one of the interpersonal motivating factors—people take pleasure in getting recognized and appreciated by other people for the sake of their personal efforts and accomplishments. The recognition principle in PSD model states that a system can increase the likelihood of its users adapting the target behavior by offering public recognition to an individual or a group [45]. For the design pattern for recognition, we present: Allow users to gain recognition by showing the system's positive feedback on their performances to their peers. Recognition can be induced in social media systems by extending the SLF pattern to elevate individual users or their performances.

Pattern Name and Classification Recognition (REC)

Intent To design software features, which enable users to gain recognition from their peers.

Motivation (Forces) When working toward a set target at times, users need a reason to focus on the target.

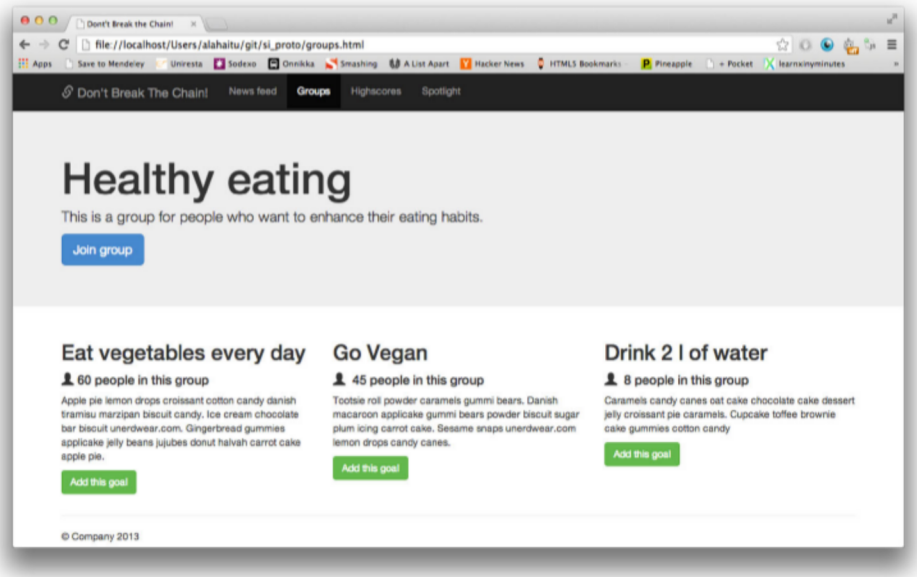

Fig. 7 Example implementation of COO pattern

Therefore, systems should provide opportunities for recognition of the top achievers. This could be on a weekly basis, for instance, and it should be for either a similar goal or one that is closely related so that the user can relate that achievement to their target behavior.

Applicability The stories about well-performing users can be presented as positive exemplars to others, as well as to give recognition and users may be given ranks or titles as they progress.

Structure The REC pattern, like the COM pattern, in our example is provided in the features of User, Chain, Goal and Story entities. Maintaining Chains that consist of Stories created at subsequent days adds to the point attribute of a User. Story itself always relates to a certain Goal. REC pattern (see Fig. 4) is implemented as the logic that promotes featured Users based on the lengths of their Chains—the number of consecutive days the user has performed a desired behavior.

Implementation Finally, the Spotlight feature (see Fig. 8) actualizes the REC pattern. To enable users to gain recognition in the system, the profiles of well-performed individuals are exhibited in the Spotlight section. The selection of the featured users can be curated by system moderators or chosen algorithmically. In our case, the spotlight page shows a snapshot of the users' profile along with their past exemplary performances within the system.

Known Uses In Fitocracy and Foursquare, for example, users' levels, achievements and titles are visible during their interactions with others.

A description of the potential user actions that summarizes the implementation section of the patterns is presented in Table 1. These are just examples of the user actions, and more tests with actual users would need to be carried to ascertain their accuracy, but we thought an outline of the potential actions would serve as a good example of what features could be implemented based on the above concepts on social influence.

5 Discussion

Persuasive software design patterns have been suggested as a new avenue of research into persuasive design [42]. Patterns are based on the premise that there rarely are entirely new problems; almost all problems—regardless of context—have been faced and solved earlier even if in a different discipline and the solutions for these can be applied in a different environment [35]. And by offering means for presenting either situations or solutions in a structured way, patterns provide a method for instantiating the design principles, in this case for social influence as presented in the PSD model [45]. Thus, they can be useful for enhancing the development of persuasive systems by providing a reusable framework that enhances standardization—of both the development process and the classification of the social influence constructs—and one not affected by iterative processes in systems design or factors that cause changes in software.

Research on persuasive software design patterns is limited, and ones with persuasive elements [e.g., 35] have not focused on actual system implementation even though it presents several patterns that are directed to social influence in its interaction, ludic and cognitive lenses. In the toolkit presented in [35], there is a peer feedback

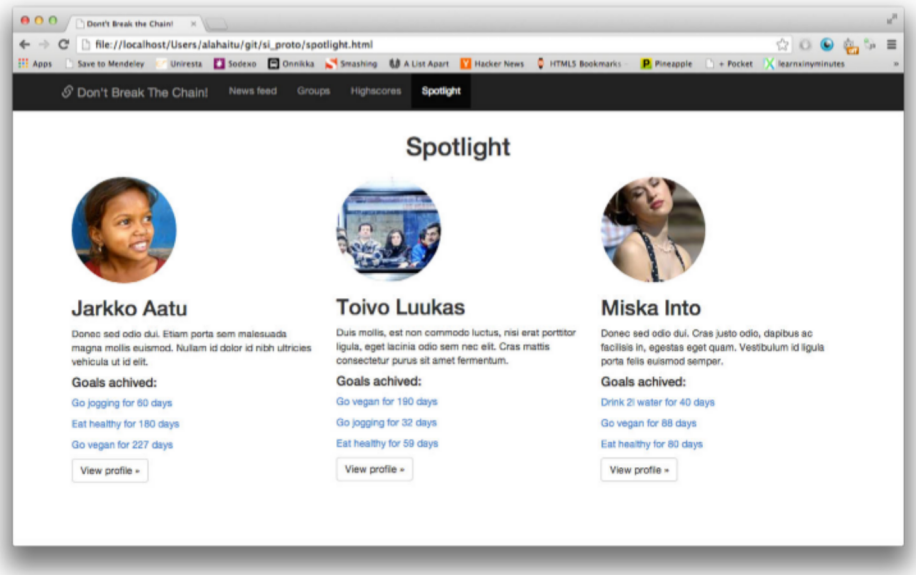

Fig. 8 Example implementation of REC software pattern

Table 1 Simplified scenarios of user interaction with the system

	User action User navigates to the site	Question(s) to consider Is the system device or platform agnostic?	System response Optimize according to device	Ideas Is the user aware of the various layouts they could use?
SLF	Browses other users shared information	What information about other users should be displayed?	User numbers and their actions displayed in a suitable manner	Show the most popular content of only those in the users close circle in the news feed
COM	Reviews the tasks and compares completed tasks to others'	Information to be displayed so as to both encourage and motivate the user?	Tasks completed and level attained by others displayed in the ranking lists	Send encouraging messages about the target behavior
COO	Checks own and the group's progress	What kind of information is the user likely to be presented with? How to encourage users to work together toward common goal?	The tasks that other users are working on are clearly visible. System shows suitable groups with similar goals to join	Provide information on how to reach similar others or those the users may like to interact with
REC	Update completed targets and duration of next task	Should the rewards automatically be visible to other users?	User gets new rewards because s/he got to a certain level	Display action steps and what is needed to attain a set target to get the next reward

function in the interaction lens and social proof and provoke empathy patterns in the cognitive lens, which suggest using SLF in design [35]. The ludic lens also contains many patterns for creating competitive interactions: scores, levels, rewards and playfulness. Reciprocation is also implied as a pattern to make users want to return a favor they have received, which facilitates cooperation [35]. Although Zuckerman and Gal-Oz [65] suggest that gamification features such as virtual rewards should be implemented with caution as they tend not to be effective because of their focus on external rewards and the system owner's goals. For long-term behavior change, the focus should be more user-centric with an emphasis on fulfilling intrinsic goals [65].

As patterns model real-world entities, we consider the user a key part in the development of the persuasive design patterns; hence, the need to associate the user attribute directly with the patterns. This helps in understanding how different users acquire knowledge due to the patterns' ability to be formulated in various ways. Our implementation of the presented software design patterns for social influence in a conceptual web service prototype suggests that they can be used to add persuasive software features to BCSSs. Of course, there is still a need to investigate and look for further evidence.

Development of prototypes based on the patterns, user testing and evaluation needs to be carried out. Prior to developing prototypes, user tests will be conducted to understand the users' opinions, perceptions and attitudes. This will primarily be based on understanding users' needs through use of user testing techniques, for example, scenarios as those in Table 1 stating potential user actions aimed at realizing their needs, and utilization techniques such as tree testing which enables the discerning of how well users interact and whether the structure of a systems is comprehensible [15]. Tree tests have been utilized to, for example, analyze the information structure of health IS [30]. After conducting user tests and analyzing the results, then systems on different problem domains and multiple software and hardware platforms could be developed and evaluated using beta-testing—distributing the initial version of the system among a set of individuals to gain their objective feedback—to gain a better understanding of the benefits and limitations of the patterns.

In their paper examining the trends in software patterns, Buschmann et al. [7] state that patterns influence developers' design and implementation of computing systems, and analysis of patterns' past, present and future trends can help developers to improve their projects. In more general, empirical BCSS research—such as applying the principles outlined above in systems development, conducting user and system testing—offers a unique opportunity for quantifying measures for system success—whether a system has achieved its intended persuasive and/or behavior change goal, and this entails clearly explaining the intent of the system, the measure(s) of success and how this has been achieved [46]. This is because research in BCSS unlike in general persuasive systems emphasizes the users' needs and goals rather than the systems features and developers' goals [50], and the ability to fully enhance the impact of persuasive systems will depend on development of features that closely align the intent of the system(s) to those of users. These features, including those for social influence can potentially increase users' motivation, because they can not only monitor their own progress, but also see how their fellow users are performing [29], and if their own goals are realized at the same time, then the potential to sustain longterm behavior change is increased.

6 Conclusion

In this article, we have proposed software design patterns for social influence, which will facilitate the development of more robust persuasive systems and provide a linkage between users' interaction with the systems and the intent of the systems designers. Firstly, the patterns are instantiations of well-known social influence features—SLF, competition, recognition, cooperation—that can be applied in actual system development. Secondly, the patterns provide a standard method for implementing the features, and they can be modeled differently when developing BCSSs and other persuasive systems that cater for different behavior(s) and/or health needs. This has paved the way for a new research track—persuasive software patterns—and implies it would be beneficial to further investigate the use of the patterns in the future research into designing persuasive systems. A natural next step would be to conduct user tests on additional prototypes derived from the patterns and evaluate the outcome of these. The patterns should also be implemented in fully functional software systems in order to get a more detailed conception of their software implementations. This type of settings would also provide an appropriate environment in which users' actions could be recorded and tracked over time.

References

1. Alahäivälä T, Oinas-Kukkonen H, Jokelainen T (2013) Software architecture design for health BCSS: case onnikka. Lecture notes in computer science, vol 7822, Persuasive Technology, pp 3–14
2. Aversano L, Canfora G, Cerulo L, Del Grosso C, Di Penta M (2007) An empirical study on the evolution of design patterns. Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Dubrovnik, Croatia. pp 385–394
3. Bandura A (1976) Social learning theory. Prentice Hall, Englewood Cliffs
4. Bandura A (1991) Social cognitive theory of self-regulation. Organ Behav Hum Decis Process 50:248–287
5. Bandura A (2001) Social cognitive theory: an agenetic perspective. Ann Rev Psychol 2001(52):1–26
6. Briñol P, Petty RE (2009) Persuasion: insights from the self-validation hypothesis. Adv Exp Soc Psychol 41:69–118
7. Buschmann F, Henney K, Schmidt DC (2007) Past, present, and future trends in software patterns. Software, IEEE 24(4):31–37
8. Chiu C, Hsu M, Wang ETG (2006) Understanding knowledge sharing in virtual communities: an integration of social capital and social cognitive theories. Decis Support Syst 42(3): 1872–1888

9. Cialdini RB, Trost MR (1998) Social influence: Social norms, conformity, and compliance. In: Gilbert D, Fiske S, Lindzey G (eds) The handbook of social psychology, vol 2, 4th edn. McGraw-Hill, New York, pp 151–192. ISBN 0070237107
10. Cialdini R (2003) Crafting normative messages to protect the environment. Curr Dir Psychol Sci 12(4):105–109
11. Cialdini R (2007) Descriptive social norms as underappreciated sources of social control. Psychometrika 72(2):263–268
12. Cialdini R (2007) Influence: the psychology of persuasion. HarperCollins Publishers, New York
13. Deterding S, Sicart M, Nacke L, O'Hara K, Dixon D (2011) Gamification. Using game-design elements in non-gaming contexts. Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA'11, 2425
14. Diaz P, Aedo I, Rosson MB (2008) Visual representation of web design patterns for end-users. Proceedings of the Working Conference on Advanced Visual Interfaces, Napoli, Italy. pp 408–411
15. Dubey SK, Rana A (2012) Analytical comparison of usability measurement methods. Int J Comput Appl 39(15):11–18
16. Fehr E, Fischbacher U (2004) Social norms and human cooperation. Trends Cognit Sci 8(4):185–190
17. Fogg BJ, Hreha J (2010) Behavior wizard: a method for matching target behaviors with solutions. Proceedings of the 5th international conference on persuasive technology, Copenhagen, Denmark, pp 117–131
18. Fogg B (2009) A behavior model for persuasive design. Proceedings of the 4th international conference on persuasive technology, Claremont, California
19. Fogg BJ (2003) Persuasive technology: using computers to change what we think and do. Morgan Kaufmann Publishers, San Francisco
20. Franch X (2013) Software requirement patterns. Proceedings of the 2013 international conference on software engineering, San Francisco, CA, USA, pp 1499–1501
21. Flynn FJ, Amanatullah ET (2012) Psyched up or psyched out? The influence of coactor status on individual performance. Organ Sci 23(2):402–415
22. Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object oriented software. Addison-Wesley, Reading
23. Gemert-Pijnen J, Wolfgang R, Langrial S, Ploderer B Oinas-Kukkonen H (2013) Expanding the research area of behavior change support systems. First International Conference on Behavior Change Support Systems, Sydney, Australia, pp 19–22
24. Gestwicki P, Sun F (2008) Teaching design patterns through computer game development. J Educ Resour Comput 8(1):2:1–2:22
25. Guadagno RE, Cialdini RB (2010) Preference for consistency and social influence: a review of current research findings. Soc Influ 5(3):152–163
26. Huotari K, Hamari J (2012) Defining gamification: a service marketing perspective. Proceedings of the 16th international academic mind trek conference, Tampere, Finland
27. Kleinrock L (1996) Nomadicity: anytime, anywhere in a disconnected world. Mob Net Appl 1:351–357
28. Kobayashi T, Saeki M (1999) Software development based on software pattern evolution. Software engineering conference, (APSEC '99) proceedings. Sixth Asia Pacific, pp 18–25
29. Langrial S, Stibe A, Oinas-Kukkonen H (2013) Practical examples of mobile and social apps using the outcome/change design matrix. First international conference on behavior change support systems, Sydney, Australia, pp 19–22
30. Le T, Chaudhuri S, Chung J, Thompson HJ, Demiris G (2014) Tree testing of hierarchical menu structures for health applications. J Biomed Inform 49:198–205
31. Leff A, Rayfield JT (2001) Web-application development using the model/view/controller design pattern. Enterprise distributed object computing conference, EDOC '01. Proceedings. Fifth IEEE International, pp 118–127
32. Lehto T, Oinas-Kukkonen H (2011) Persuasive Features in web-based alcohol and smoking interventions: a systematic review of the literature. J Med Internet Res 13:3
33. Zhu L, Babar MA, Jeffery R (2004) Mining patterns to support software architecture evaluation. Software architecture, WICSA 2004. Proceedings. Fourth working IEEE/IFIP conference on, pp 25–34
34. Lindqvist J, Cranshaw J, Wiese J, Hong J, Zimmerman J (2011) I'm the mayor of my house: Examining why people use foursquare: a social-driven location sharing application. Proceedings of the SIGCHI Conference on Human factors in computing systems, Vancouver, BC, Canada. 2409–2418. doi:10.1145/1978942.1979295
35. Lockton D (2013) Design with intent: a design pattern toolkit for environmental and social behaviour change. Brunel University School of Engineering and Design PhD Theses
36. Lockton D (2012). Persuasive technology and digital design for behaviour change. Retrieved on 15 Dec 2013 from http://ssrn.com/abstract=2125957
37. Lustria MLA, Cortese J, Noar SM, Glueckauf RL (2009) Computer-tailored health interventions delivered over the web: review and analysis of key components. Patient Educ Couns 74(2):156–173
38. Malone TW, Lepper M (1987) Making learning fun: a taxonomy of intrinsic motivations for learning. In: Snow RE, Farr MJ (eds) Aptitude, learning and instruction: III. Conative and affective process analyses. Erlbaum, Hillsdale, NJ, pp 223–253
39. Mavrodiev P, Tessone CJ, Schweitzer F (2013) Quantifying the effects of social influence. Nature Sci Rep 3:1360
40. Nass C, Steuer J, Tauber ER (1994) Computers are social actors. Proceedings of the SIGCHI conference on human factors in computing systems, Boston, Massachusetts, USA, pp 72–78
41. Nolan JM (2011) The cognitive ripple of social norms communications. Group Process Intergroup Relat 14(5):689–702
42. Oinas-Kukkonen H (2010) Behavior change support systems: a research model and agenda. Lecture notes in computer science, Persuasive, vol. 6137, Springer, Keynote Paper, pp 4–14
43. Oinas-Kukkonen H (2013) A foundation for the study of behavior change support systems. Personal and ubiquitous computing, vol. 17, No. 6, pp 1223–1235
44. Oinas-Kukkonen H, Harjumaa M (2008) Towards deeper understanding of persuasion in software and information systems, Proceedings of the 1st international conference on advances in computer-human interaction, pp 200–205
45. Oinas-Kukkonen H, Harjumaaa M (2009) Persuasive systems design: key issues, process model, and system features. Commun Assoc Infor Sys 24(28):485–500
46. Oinas-Kukkonen H, Oinas-Kukkonen H (2013) Humanizing the web: change and social innovation. Palmgrave Macmillan, Basingstoke
47. Peña-Mora F, Vadhavkar S (1997) Augmenting design patterns with design rationale, artificial intelligence for engineering design, analysis and manufacturing (AIEDAM). Special Issue on Design Rationale, vol 11, pp 93–108
48. Petty RE, Cacioppo JT (1986) The elaboration likelihood model of persuasion. In: Berkowitz L (ed) Advances in experimental social psychology, vol 19. Academic Press, New York, pp 123–205
49. Thung PL, Ng CJ, Thung SJ, Sulaiman S (2010) Improving a web application using design patterns: a case study. Int Symp Infor Technol (ITSim) 1:1–6

50. Ploderer B, Reitberger W, Oinas-Kukkonen H, Gemert-Pijnen J (2014) Social interaction and reflection for behavior change. Pers Ubiquit Comput. doi:10.1007/s00779-014-0779-y (this issue)
51. Pribik I, Felfernig A (2012) Towards persuasive technology for software development environments: an empirical study. Persuasive technology. Design for health and safety. Lect Notes Comput Sci 7284(2012):227–238
52. Rosas P, Howard S, Gibbs M (2013) Managing multiple influences: self-monitoring and social comparison at the same time and context. First international conference on behavior change support systems, Sydney, Australia, pp 19–22
53. Smith JR, Louis WR, Schultz PW (2011) Introduction: social influence in action. Group Process Intergroup Relat 14(5):599–603
54. Stibe A, Oinas-kukkonen H (2012). Exploring the effects of social influence on user behavior targeted to feedback sharing. Proceedings of the IADIS WWW/Internet 2012 conference (ICWI 2012), pp 281–289
55. Stibe A, Oinas-Kukkonen H, Lehto T (2013) Exploring social influence on customer engagement: a pilot study on the effects of social learning, social comparison, and normative influence. 2013 46th Hawaii International Conference on System Sciences, pp 2735–2744
56. Taleb M, Seffah A, Abran A (2007) Patterns-oriented design applied to cross-platform web-based interactive systems. IEEE international conference on information reuse and integration, IRI 2007, pp 122–127
57. Trese T, Tilley S (2007) Documenting software systems with views V: Towards visual documentation of design patterns as an aid to program understanding. Proceedings of the 25th annual ACM international conference on design of communication, El Paso, Texas, USA, pp 103–112
58. Ning W, Liming L, Yanzhang W, Yi-bing W, Jing Wang (2008) Research on the web information system development platform based on MVC design pattern. *WI-IAT '08*. Proc IEEE WIC ACM Int Conf Web Intell Intell Agent Technol 3:203–206
59. Weiksner GM, Fogg BJ, Liu X (2008) Six patterns for persuasion in online social networks. Proceedings of the 3rd international conference on persuasive technology, Oulu, Finland, pp 151–163
60. Wiafe I, Nakata K, Gulliver S (2014) Categorizing users in behavior change support systems based on cognitive dissonance. Pers Ubiquit Comput. doi:10.1007/s00779-014-0782-3
61. Wood W (2000) Attitude change: persuasion and social influence. Annu Rev Psychol 51:539–570
62. Zajonc RB (1965) Social facilitation. Science 149:269–274
63. Zeal J, Smith SP, Scheepers R (2012) Revisiting social influence in the ubiquitous computing era. System science (HICSS), 2012 45th Hawaii international conference on, pp 889–898
64. Zhu Z (2009) Study and application of patterns in software reuse. Control, automation and systems engineering, 2009. CASE 2009. IITA international conference on, pp 550–553
65. Zuckerman O, Gal-Oz A (2014) Deconstructing gamification: evaluating the effectiveness of continuous measurement, virtual rewards, and social comparison for promoting physical activity. Pers Ubiquit Comput. doi:10.1007/s00779-014-0783-2